Ancestral Benders' Cuts and Multi-term Disjunctions for Mixed-Integer Recourse Decisions in Stochastic Programming *

Yunwei Qi†, Suvrajeet Sen†‡

qi.47@osu.edu, s.sen@usc.edu

September 3, 2013

Abstract

This paper focuses on solving two-stage stochastic mixed integer programs (SMIPs) with general mixed integer decision variables in both stages. We develop a decomposition algorithm in which the first stage approximation is solved using a branch-and-bound tree with nodes inheriting Benders' cuts that are valid for their ancestor nodes. In addition, we develop two closely related convexification schemes which use multi-term disjunctive cuts to obtain approximations of the second stage mixed-integer programs. We prove that the proposed methods are finitely convergent. One of the main advantages of our decomposition scheme is that we use a Benders-based branch-and-cut approach in which linear programming approximations are strengthened sequentially. Moreover as in many decomposition schemes, these subproblems can be solved in parallel. We also illustrate these algorithms using several variants of an SMIP example from the literature, and present preliminary evidence of the scalability of these algorithms as the number of scenarios increases.

Keywords: Two-stage stochastic mixed-integer programs, cutting plane tree algorithm, multi-term disjunctive cut, Benders' decomposition.

1 Introduction

Stochastic mixed-integer programs (SMIPs) have long been recognized as an important class of models for many practical operational problems (see e.g. [Wollmer, 1980]). However, algorithmic advances to solve SMIP models have lagged behind other forms of stochastic programs (SP). In addition to the standard difficulties associated with stochastic linear programming (e.g. designing scalable ways to approximate

*This work is supported in part by NSF-CMMI Grant 1100383 and AFOSR Grant FA9550-13-1-0015

†Integrated Systems Engineering, The Ohio State University,Columbus OH, 43210, USA

‡Epstein Department of Industrial and Systems Engineering, University of Southern California, Los Angeles CA, 90089-0193, USA

the expected recourse/value function), SMIP formulations with mixed-integer recourse decisions in the second stage encounter value functions that are possibly non-convex and discontinuous. Early work by [Carøe and Tind, 1998] presented a decomposition algorithm based on mixed integer programming (MIP) duality, and while it is conceptually applicable to SMIP with general mixed-integer recourse decisions, the algorithm is not easily realizable because it requires calculations involving exact MIP value functions. Such value functions are not only difficult to construct in the second stage, but also lead to discontinuous and non-convex first stage approximations in general. Other decomposition algorithms, based on scenario decomposition, were proposed in [Carøe and Schultz, 1997] and [Lulli and Sen, 2004]. These algorithms essentially view the SMIP problem as a very large scale MIP using a deterministic equivalent formulation. The decomposition principles used by the above methods are dual to each other (price and resource directive decomposition respectively) and may be recommended for instances in which special structures associated with scenario subproblems can be exploited (as in unit-commitment models, lot-sizing models etc.). However, when the number of scenarios is very large, and special structures are either absent or difficult to exploit, such scenario decomposition methods are not particularly effective.

Subsequent to the work of [Carøe and Tind, 1998], most authors addressed some sub-class of the two-stage SMIP problem. For instance, the global optimization algorithm of [Ahmed et al., 2004] assumed fixed tenders (i.e. deterministic T matrix in (2) below) in the two-stage model. Others have addressed alternative sub-classes which either focus on mixed-binary recourse decisions, or pure integer recourse decisions. Decomposition-based cutting plane algorithms (e.g. [Sen and Higle, 2005], [Sherali and Zhu, 2006], [Ntaimo, 2010]), novel branch-and-bound methods ([Escudero et al., 2007]), decomposition based branch-and-cut ([Sen and Sherali, 2006]), as well as primal approaches using certain IP value function characterizations ([Kong et al., 2006], [Trapp et al., 2012]) now offer a range of algorithms for alternative model characteristics. As a consequence of the sharper focus, there has been significant progress with SMIP algorithms for specialized cases of SMIP. In contrast to some of the earlier multi-stage SMIP algorithms, these methods are based on time-staged decomposition, very much in the spirit of Benders' decomposition. We refer to surveys by ([Louveaux and Schultz, 2003]) and ([Sen, 2010]) for discussions related to these and related advances.

In this paper, we are interested in designing time-staged decomposition algorithms for solving two-stage SMIPs in which mixed-integer decisions appear in both stages. In other words, we return to the class of models addressed in [Carøe and Tind, 1998]. Fortunately, due to significant algorithmic advances in the interim, we are able to draw upon new approximations that will not only ensure finite convergence, but also avoid intractable operations in each iteration. Using nodes of a first stage branch-and-bound tree to hold "genetic markers", our algorithm will couple conditionally valid Benders' cuts and second stage disjunctive cuts to create computable approximations which can solve general SMIP problems in

finitely many iterations. This also contrasts with Benders' decomposition and related SMIP algorithms ([Sen, 2010]) in which the approximations are globally valid.

The SMIP formulation we consider is stated in (1) and (2).

$$\min_{x \in X \cap Q_1} c^{\mathsf{T}} x + \mathsf{E}[f(x, \tilde{\omega})] \tag{1}$$

where

$$X = \{x \mid Ax \leq b, x_i \text{ is integer, for } \forall i \in I_2 \subseteq I_1 = \{1...n_1\}\}$$
$$Q_1 = \{x \mid l_1 \leq x \leq u_1\}.$$

We require the objective coefficient vector $c \in \mathbb{R}^{n_1}$, the constraint matrix $A \in \mathbb{R}^{m_1 \times n_1}$ and the right hand side $b \in \mathbb{R}^{m_1}$. Also $\tilde{\omega}$ denotes a random variable, and for each scenario (realization) ω of $\tilde{\omega}$, we define the recourse function by

$$\begin{aligned} f(x, \omega) = \min \quad & g(\omega)^{\top} y \\ \text{s.t.} \quad & W(\omega) y \geq r(\omega) - T(\omega) x \\ & y \in Y \cap Q_2 \end{aligned} \tag{2}$$

where

$$Y = \{y \mid y_j \text{ is integer, for } \forall j \in J_2 \subseteq J_1 = \{1...n_2\}\} \subseteq \mathcal{R}^{n_2}$$
$$Q_2 = \{y \mid l_2 \leq y \leq u_2\}.$$

For subproblem (2), the decision variable $y \in \mathbb{R}^{n_2}$, objective coefficient $g(\omega) \in \mathbb{R}^{n_2}$, constraint matrix $W(\omega) \in \mathbb{R}^{m_2 \times n_2}$, $r(\omega) \in \mathbb{R}^{m_2}$ and $T(\omega) \in \mathbb{R}^{m_2 \times n_1}$. We assume the random variable $\tilde{\omega}$ is discrete with each scenario ω having a non-zero probability $p(\omega)$, for all $\omega \in \Omega$.

The algorithms that we propose in this paper will impose the following assumptions on the model.

A1 Both X and Y are assumed to be non-empty mixed-integer sets, and the integer variables in both stages are bounded.

A2 The random variable $\tilde{\omega}$ in the problem is discrete, with a finite number of scenarios, $\omega \in \Omega$, each with an associated non-zero probability of occurrence $p(\omega), \forall \omega \in \Omega$.

A3 For any $x \in X \cap Q_1$, the set defined by $\{y \mid W(\omega)y \geq r(\omega) - T(\omega)x, y \in Y \cap Q_2\}$ is feasible for all $x \in X \cap Q_1$ and all $\omega \in \Omega$.

Due to assumption **A2**, (1) can be rewritten as:

$$\min_{x \in X \cap Q_1} c^{\mathsf{T}} x + \sum_{\omega \in \Omega} p(\omega) f(x, \omega). \tag{3}$$

For the rest of the paper, we begin by first introducing the overall architecture of an algorithm in which a B&B algorithm in the first stage will control how approximations are created and passed from one generation of nodes to another. Subsequently, we present two closely related convexification schemes for the second stage, one based on the cutting plane tree method ([Chen et al., 2012]), and the other referred to as a branch-and-bound (B&B)-based convexification method. Both of these methods can be used to *approximate* the second stage value function $f(x, \omega)$. Finally, several variants of an SMIP example in the literature illustrate how our approach can be used under different assumptions (or structures) for SMIP models. In addition we present preliminary evidence that the decomposition framework promises to be more scalable than solving a deterministic equivalent problem (3) using a standard commercial MIP solver. Overall, our framework provides the most comprehensive time-staged decomposition approach to date, allowing randomness in all data elements, while also allowing general mixed-integer variables as decision variables in both stages.

2 An Algorithm with Ancestral Benders' Cuts

If we denote decision variables y under scenario ω as $y(\omega)$, the deterministic equivalent formulation (DEF) formulation for (3) is

$$\min_{x, y(\omega)} \quad c^{\top} x + \sum_{\omega \in \Omega} p(\omega) \sum g(\omega)^{\top} y(\omega) \tag{4a}$$

$$\text{s.t.} \quad T(\omega) x + W(\omega) y(\omega) \geq r(\omega), \forall \omega \in \Omega \tag{4b}$$

$$x \in X \cap Q_1, y(\omega) \in Y \cap Q_2. \tag{4c}$$

When x is restricted to be binary, the resulting first stage problems may be classified as facial disjunctive programs [Balas, 1979]. In the context of two-stage binary SIP, [Sherali and Fraticelli, 2002] showed that solving problem (4) is equivalent to solving (5) provided $x = \bar{x}$ is facial with respect to Q_1.

$$\min_{x \in X} \quad c^\top x + \sum_{\omega \in \Omega} p(\omega) \sum g(\omega)^\top y(\omega) \tag{5a}$$

$$\text{s.t.} \quad (x, y(\omega)) \in \mathbf{conv}\{(x, y(\omega)) | T(\omega)x + W(\omega)y(\omega) \geq r(\omega),$$

$$x \in Q_1, y \in Y \cap Q_2\} \forall \omega \in \Omega \tag{5b}$$

$$x \in X \cap Q_1, \tag{5c}$$

When the vectors $(x, y(\omega))$ are allowed to be mixed-integer, two issues arise in the context of decomposition algorithms: (1) in the presence of general integers, the projection of the feasible set for $x = \bar{x}$ may not be facial with respect to Q_1, and (2) the presence of mixed-integer variables in the second stage presents the possibility that traditional cutting plane methods (e.g. disjunctive cuts and Gomory cuts) used in [Sen and Higle, 2005, Sherali and Fraticelli, 2002, Gade et al., 2012] may fail to converge in the second stage. For these reasons we will design a branch-and-bound (B&B) algorithm for the first stage model, and each node of the first stage B&B tree will be coupled to the second stage by using Benders' decomposition. A key feature of this algorithm is to allow every node of the first stage B&B tree to inherit Benders' cuts produced previously by its ancestral nodes, but is allowed to proceed adding more focused cuts which are only valid within the range specific to the node. We will refer to this procedure as the "Ancestral Benders' Cut" (**ABC**) Algorithm. This algorithm allows us to integrate three important ingredients which are also important for its convergence. They are:

a) The B&B process of the first stage will ultimately construct partitions (nodal instances) which will satisfy the facial property (as required by [Sherali and Fraticelli, 2002]).

b) Ideas from disjunctive decomposition [Sen and Higle, 2005] will ensure that any second stage instance whose first stage input x satisfies the facial property will, provide a Benders' cut whose value agrees with the objective value $f(x, \omega)$ for all $\omega \in \Omega$.

c) Multi-term disjunctions that were developed as part of the Cutting Plane Tree (CPT) algorithm (see [Chen et al., 2011]) can be used to approximate the convex hull of mixed-integer sets in the second stage.

In describing the algorithm, we use index k to denote an iteration index. A B&B tree for the first stage provides a partition of the bounding constraints denoted Q_1. In iteration k, we let $t(k)$ denote a node of the first stage B&B tree such that the first stage solution $x^k \in Q_1^{t(k)}$, where $Q_1^{t(k)} \subseteq Q_1$ is the

bounding constraints of node $t(k)$. Next define

$$\mathcal{Y}(Q_1^{t(k)}, \omega) \equiv \{(x, y(\omega)) \mid T(\omega)x + W(\omega)y \geq r(\omega), x \in X \cap Q_1^{t(k)}, y \in Y \cap Q_2\}. \tag{6}$$

In addition to node $t(k)$'s bounds $Q_1^{t(k)}$, $\mathcal{Y}(Q_1^{t(k)}, \omega)$ covers all constraints for variable x and $y(\omega)$. Because X and Y contain mixed-integer points, to evaluate the value function for each scenario ω, one might consider approximating the convex hull of $\mathcal{Y}(Q_1^{t(k)}, \omega)$ (denoted as $\mathbf{conv}\{\mathcal{Y}(Q_1^{t(k)}, \omega)\}$). One way is to construct an approximation motivated by the CPT algorithm ([Chen et al., 2011]) which helps to develop a polyhedral relaxation $\mathcal{Y}_L(Q_1^{t(k)}, \omega)$ such that

$$\mathcal{Y}_L(Q_1^{t(k)}, \omega) \supseteq \mathbf{conv}\{\mathcal{Y}(Q_1^{t(k)}, \omega)\}. \tag{7}$$

Moreover, for fixed $x^k \in \mathbf{vert}(X \cap Q_1^{t(k)})$, the polyhedral approximation $\mathcal{Y}_L(Q_1^{t(k)}, \omega)$ generated by CPT algorithm ([Chen et al., 2011]) ensures that $f^t(x^k, \omega) = f_L^{t,k}(x^k, \omega)$ where $f^t(x, \omega)$ is defined as

$$f^t(x, \omega) = f(x, \omega) \text{ for } x \in Q_1^t. \tag{8}$$

and $f_L^{t,k}(x, \omega)$ as

$$f_L^{t,k}(x, \omega) = \min\{g(\omega)^\top y : \ (x, y) \in \mathcal{Y}_L(Q_1^{t(k)}, \omega)\}. \tag{9}$$

For brevity, we will use $f(x^k, \omega)$ instead of $f^t(x^{t(k)}, \omega)$ and $f_L^k(x^k, \omega)$ instead of $f_L^{t,k}(x^{t(k)}, \omega)$ in the rest of the paper. Thus appealing to the CPT procedure, one may construct a polyhedral approximation which may be described as follows,

$$\mathcal{Y}_L(Q_1^{t(k)}, \omega) = \{(x, y) \mid T(\omega)x + W(\omega)y \geq r(\omega), \Pi_1^{t(k)}(\omega)x + \Pi_2^{t(k)}(\omega)y \geq \Pi_0^{t(k)}(\omega), x \in X \cap Q_1^{t(k)}, y \geq 0\}, \tag{10}$$

where

$$\Pi_1^{t(k)}(\omega)x + \Pi_2^{t(k)}(\omega)y \geq \Pi_0^{t(k)}(\omega) \tag{11}$$

denote the collection of inequalities that are valid for $x \in X \cap Q_1^{t(k)}, y \in Y \cap Q_2$.

Note that unlike the case of binary first stage variables, (11) may not be globally valid, and may, in fact, delete feasible points (x, y) with $x \in X \cap \{Q_1 \backslash Q_1^{t(k)}\}, y \in Y \cap Q_2$. However, by associating valid inequalities with specific subsets Q_1^t, we are able to obtain valid lower bounds for any specific subset Q_1^t. As the B&B process becomes more focused, the cut enhanced approximations provide tighter lower bounds. We will return to discuss this process, but first we present a proposition that motivates the approximations.

Proposition 1. *For any $\omega \in \Omega$ and fixed $x = \bar{x} \in X \cap Q_1^{t(k)}$, we have $f(x,\omega) \geq f_L^k(x,\omega)$. Furthermore, if $\bar{x} \in \mathbf{vert}(X \cap Q_1^{t(k)})$, where $\mathbf{vert}(X \cap Q_1^{t(k)})$ denotes the vertices of $X \cap Q_1^{t(k)}$, we have $f(x,\omega) = f_L^k(x,\omega)$.*

Proof. Since $\mathcal{Y}_L(Q_1^{t(k)},\omega) \supseteq \mathcal{Y}(Q_1^{t(k)},\omega)$, we have $f(x,\omega) \geq f_L^k(x,\omega)$. If $\bar{x} \in \mathbf{vert}(X \cap Q_1^{t(k)})$, the restriction $x = \bar{x}$ is facial respect to $\mathcal{Y}_L(Q_1^{t(k)},\omega))$. Hence following ([Sherali and Fraticelli, 2002]), we have $f(x,\omega) = f_L^k(x,\omega)$. □

Proposition 1 helps us derive the decomposition algorithm for a general SMIP problem. For fixed x^k and $Q_1^{t(k)}$, the second stage convexification process will yield the set $\mathcal{Y}_L(Q_1^{t(k)},\omega)$. Let the matrix $W^k(\omega)$ denote a matrix consisting of the original matrix $W(\omega)$ together with the entire collection of cuts generated for scenario ω, and let $W^{t(k)}(\omega)$ denote the subset of inequalities in the matrix $W^k(\omega)$ which are valid for $x \in Q_1^{t(k)}$. Thus we put $W^{t(k)}(\omega) = \begin{bmatrix} W(\omega) \\ \Pi_2^{t(k)}(\omega) \end{bmatrix}$, and similarly, $r^{t(k)}(\omega) = \begin{bmatrix} r(\omega) \\ \Pi_0^{t(k)}(\omega) \end{bmatrix}$, $T^{t(k)}(\omega) = \begin{bmatrix} T(\omega) \\ \Pi_1^{t(k)}(\omega) \end{bmatrix}$. Then a relaxation of the integer recourse (or value) function can be obtained by solving the following,

$$f_L^k(x^k,\omega) = \min \quad \sum g(\omega)^\top y \tag{12a}$$

$$\text{s.t.} \quad W^{t(k)}(\omega)y \geq r^{t(k)}(\omega) - T^{t(k)}(\omega)x^k \tag{12b}$$

$$y \geq 0. \tag{12c}$$

Unlike standard Benders' decomposition (which is normally applied to the case where the second stage is a linear program), the subproblem stated above is a relaxation of the MIP subproblem. Now, if we wish to derive a subgradient of the value function f_L^k, then, we can adopt the Benders' procedure for a restricted LP, and adding the resulting Benders' cut to previously generated cuts, will yield a progressively improving approximation of the actual IP value function. Let $\Theta^{t(k)}(\omega)$ denote the optimal dual multipliers associated with (12). Then the Benders' cut based on approximation (12) is given as

$$\eta^{t(k)}(\omega) \geq \Theta^{t(k)}(\omega)^\top (r^{t(k)}(\omega) - T^{t(k)}(\omega)x). \tag{13}$$

Such a Benders' cut for the master problem can be written as

$$\eta_t \geq \xi_k - {\zeta_k}^\top x \text{ for } x \in Q_1^{t(k)} \tag{14}$$

where

$$\begin{aligned}\xi_k &= \sum_{\omega\in\Omega} p(\omega)\Theta^{t(k)}(\omega)^\top r^{t(k)}(\omega)\\ \zeta_k &= \sum_{\omega\in\Omega} p(\omega)\Theta^{t(k)}(\omega)^\top T^{t(k)}(\omega).\end{aligned} \tag{15}$$

Since such a cut is valid for only those first stage decisions such that $x \in Q_1^{t(k)}$, accordingly, they will be used for the B&B node $t(k)$, as well as its descendant nodes. That is, the Benders' cuts generated for Q_1^t can be used for all subsets $Q_1' \subseteq Q_1^t$. Similar inheritance also happens for second stage approximation (12b) or more specifically the cuts in (11). These cuts, generated for a particular first stage B&B node $t(k)$ can also be used for descendant nodes of $t(k)$. One could treat these inheritance properties in a manner analogous to the inheritance of chromosomes in biology. So we use a binary sequence denoted as $\mathcal{G}^t = [\mathcal{G}_x^t, \mathcal{G}_y^t]$ to encode valid cuts for node t where $\mathcal{G}_x^t$ is for Benders' cuts and $\mathcal{G}_y^t$ is for second stage approximation cuts (i.e. CPT cuts), analogous to X,Y chromosome. Each bit of the sequence can only have value 1 or 0, where "1" indicates that the cut is valid for node t, and "0" indicates that it is not. The initial sequence for node t is the same as its parent node. When a new valid cut is derived for node t, one extends either $\mathcal{G}_x^t$ or $\mathcal{G}_y^t$ depending on whether it is a Benders' cut or a second stage approximation cut. The extended code is set to 1 for node t (and descendants), whereas all other nodes have an extended sequence with its corresponding bit set to 0.

We now proceed to discuss the B&B method for the first stage. Suppose the set of active (unfathomed) nodes for the first stage is denoted as $\mathcal{T}_1^k$, and let Q_1^t denote the bounding constraints for $t \in \mathcal{T}_1$. $\mathcal{G}_x^t[s]$ extracts the bit value for s-th Benders' cut. Then the lower bounding master problem for node t at iteration k is as follows,

$$\min \quad c^\top x + \eta_t \tag{16a}$$

$$\text{s.t.} \quad \eta_t \geq \xi_s - \zeta_s x, \text{for all } s \text{ such that } \mathcal{G}_x^t[s] = 1 \tag{16b}$$

$$\eta_t \geq -M, \tag{16c}$$

$$x \in X_L \cap Q_1^t, \tag{16d}$$

where $-M$ is a valid lower bound on second stage value function, and

$$X_L = \{x \mid Ax \leq b\} \subseteq \mathcal{R}^{n_1}. \tag{17}$$

The entire algorithm starts from iteration $k = 1$ with $\mathcal{T}_1^k = \{o\}$ where the root node o has bounding

constraint $Q_1^o \equiv Q_1$ and we initialize $\mathcal{G}^t$ to be NULL. At iteration k, problem (16) is solved for each $t \in \mathcal{T}_1^k$ and for our breadth-first approach, we choose the node with the least lower bound as the node to branch on. Suppose this node is $\bar{t}$. Following a variable selection rule such as choosing the fractional variable with smallest index or the most relative fractional variable as shown in (18) and (19),

$$\theta_i = \min\{x_i^t - l_{1i}^t, u_{1i}^t - x_i^t\} \tag{18}$$

$$p \in \text{argmax}_{i=1,\ldots,n}\{\frac{\theta_i}{u_{1i}^t - l_{1i}^t}\}, \tag{19}$$

we can select variable x_p and split its bounding constraint $[l_{1p}, u_{1p}]$ into (20).

$$[l_{1p}, \lfloor x_p \rfloor] \text{ and } [\lceil x_p \rceil, u_{1p}], \text{ if } p \in I_2 \tag{20}$$

The sequence $\mathcal{G}^{\bar{t}}$ which contains Benders' cut (16b) for $\bar{t}$, is now used to initialize codes for two new nodes. Again the node with least lower bound among $\mathcal{T}_1^k$ is selected and this process repeats until the mixed-integer optimum is found.

When a solution to the master problem is found (denoted as x^k), we identify the first stage node to which x^k belongs, and refer to it as $t(k)$ and its bounding constraint as $Q_1^{t(k)}$. Given x^k and $Q_1^{t(k)}$, we approximate the second stage value function for all ω as described in (14). With value function encoded in $\mathcal{G}^{t(k)}$ updated, the master problem continues to find new integer solutions and updating nodal value functions until the node is fathomed, or the algorithm stops. A summary of the prescribed process is shown in Algorithm 1, and we refer to it as the **ABC** (for Ancestral Bender's Cut) algorithm .

Proposition 2. *Assume that the process of deriving (14) is finite, and let x^k denote a first stage solution such that $x^k \in \{\arg\min E[f_L^k(x,\omega)] : x^k \in X_L \cap Q_1^t\}$. Suppose that for any t, there exists a finite iteration $K(t)$ such that for $k \geq K(t)$ we either have $v^{t(k)} \geq V$ or $f_L^k(x^k,\omega) = f(x^k,\omega)$. Then, the B&B procedure produces an optimal solution x^*, as well as its optimal value V^* in finitely many iterations.*

Proof. Since the range of first stage variables is finite, there can only be finitely many partitions. In addition, those sets that are fathomed (either because of feasibility or because the lower bound exceeds an upper bound) satisfy $E[f_L^k(x,\omega)] \geq V \geq V^*$, $\forall x \in X_L \cap Q_1^t$. Moreover, the hypothesis ensures that any partition can only have $f_L^k(x^t,\omega) < f(x^t,\omega)$ for finitely many iterations. Consequently after finitely many iterations, $v^t = E[f_L^k(x^t,\omega)] = E[f(x^t,\omega)]$. Because the B&B process must ultimately satisfy the MIP restrictions of the first stage, the previous equation actually implies that $v^t = E[f_L^k(x^t,\omega)] = E[f(x^t,\omega)] \geq V^*$ for all those t for which x^t satisfy the first stage MIP restrictions. Hence we must have at least one index t for which $v^t = V^*$, implying that $x^t \equiv x^*$, which implies that the **ABC algorithm**

Algorithm 1 Ancestral Benders' Cut (ABC) Algorithm

Initialize: Set iteration $k = 1$, objective value upper bound $V = \infty$, first stage active nodes $\mathcal{T}_1^k = \{o\}$ with $Q_1^o \equiv \{x | l_1 \leq x \leq u_1\}$. Set $\mathcal{G}^t = \emptyset$. Let ϵ denote the stopping tolerance and $(x^*, y^*(\omega))$ for $\omega \in \Omega$ the incumbent solutions. Solve problem (16) at node o and get its lower bound v^o, and set the global lower bound $v = v^o$.

while true **do**

Denote the node with the least lower bound as $t(k)$, and update the global lower bound:

$$v \leftarrow \min_{t \in \mathcal{T}_1^k} \{v^t\}.$$

if $V - v \leq \epsilon$ **then STOP**.

end if

if $x_i^{t(k)}$ for $i \in I_2$ are integers **then**

Derive (14) and update $\mathcal{G}^{t(k)}$.

Update V and incumbent $(x^*, y^*(\omega))$ if $y^k(\omega)$ satisfies mixed-integer restrictions for all $\omega \in \Omega$.

Update v^t for $t \in \mathcal{T}_1^k$ by re-solving problem (16) with updated $\mathcal{G}^{t(k)}$.

else

Choose variables to split and add two new child nodes $t^1(k)$,$t^2(k)$ of node $t(k)$ as in (20).

Solve problem (16) for both new nodes and obtain $v^{t^1(k)}$ and $v^{t^2(k)}$.

end if

Fathom nodes for which $v^t \geq V$, the upper bound V: $\mathcal{T}_1^{k+1} \leftarrow \mathcal{T}_1^k \setminus \{t \mid t \in \mathcal{T}_1^k, V - v^t \geq \epsilon\}$.

$k \leftarrow k + 1$

end while

terminates in finitely many steps. □

In the following section, we show how one satisfies the requirement in the above proposition that objective function evaluations should become more accurate as the search descends into the leaf nodes of the B&B tree.

3 Successive Value Function Approximations using Multi-term Disjunctions

In order to ensure an effective decomposition algorithm, objective function approximations generated in one iteration should be re-usable in subsequent iterations. This philosophy was adopted in designing the D^2 algorithm ([Sen and Higle, 2005]) where disjunctive cuts defining the convex approximation of $Y \cap Q_2$ were made reusable for subsequent iterations by convexifying the function defining the right hand side of the cuts as a relatively simple function of the first stage decision x. The promising results from the D^2 algorithm ([Yuan and Sen, 2009]) for binary SMIPs motivates us to extend that algorithm to more general cases considered in this paper.

We will discuss two approaches for approximating the second stage value function. Both methods will convexify the set of feasible solutions of the second stage. However, one of these will be based on a pure cutting plane approach, while the other will perform a convexification prompted by a B&B tree.

3.1 Convexification using Pure Cutting Planes

Until recently, pure cutting plane algorithms were not known to provide finitely convergent algorithms for MIP problems with general mixed-integer variables. For example, ([Balas, 1979], [Owen and Mehrotra, 2001]) have shown that for general MIP, traditional two-term disjunctive cuts (e.g.[Balas et al., 1993]) are inadequate to the task of satisfying the conditions of Proposition 2. In a recent paper, [Chen et al., 2011] have shown that by using cutting planes resulting from multi-term disjunctions, it is possible to obtain a finitely convergent algorithm for MIP problems with general mixed-integer variables. Thus, such a method will be able to deliver the property required by Proposition 2. Because these multi-term disjunctions are intimately tied to the Cutting Plane Tree (CPT) algorithm, we will refer to the resulting cuts as "CPT cuts".

Suppose that at the k-th iteration of **ABC** algorithm, we have a fixed $x^k \in X \cap Q_1^{t(k)}$ and an initial approximation $f_L^k(x,\omega)$ from previous iterations' solve $f_L^{k-1}(x,\omega)$. We seek to approximate $f(x,\omega)$ further using x^k and $f_L^k(x,\omega)$ by executing the cutting plane tree (CPT) algorithm for a few iterations. Note that this approximation is only valid for $x \in X \cap Q_1^{t(k)}$. To initialize a sequence of subproblem approximations of the CPT algorithm, we start by solving the subproblem LP relaxation $f_L^k(x,\omega)$ (as shown in (21)) with $x = x^k$.

$$\begin{aligned} f_L^k(x,\omega) = \min \quad & g(\omega)^\top y \\ \text{s.t.} \quad & W^{t(k)}(\omega) y \geq r^{t(k)}(\omega) - T^{t(k)}(\omega) x \\ & y \in Y_L \cap Q_2 \\ & Y_L = \{y \geq 0, y \in \mathcal{R}^{n_2}\} \end{aligned} \tag{21}$$

In general, this relaxation does not yield a mixed-integer feasible point, and we build approximations of the mixed-integer set using the CPT cuts which delete non-integer solutions as we encounter them. Let d denote the iteration counter of the CPT algorithm (for the second stage), and let $y^d(\omega)$ denote a solution with some fractional variable(s) at iteration d. Let $\mathcal{T}_2^d$ denote an index set of the sets that constitute a partition $\{Q_2^t, t \in \mathcal{T}_2^d\}$ of Q_2. For each $t \in \mathcal{T}_2^d$, Q_2^t denote bounding constraints for the vectors y as shown in (22).

$$Q_2^t = \{x | l_2^t \leq y \leq u_2^t\} \text{ for } \forall t \in \mathcal{T}_2^d \tag{22}$$

The tree $\mathcal{T}_2^d$ embodies a disjunctive relaxation of $Y \cap Q_2$ because we require the following condition to hold,

$$\bigcup_{t \in \mathcal{T}_2^d} (Y_L \bigcap Q_2^t) \supseteq Y \cap Q_2. \tag{23}$$

Since $y^d(\omega)$ is presumed to have some fractional values for integer variables, we will delete the point $(x^k, y^d(\omega))$ from the higher dimensional polytope. Thus, we will construct a disjunctive set which satisfies the following,

$$(x^k, y^d(\omega)) \notin \bigcup_{t \in \mathcal{T}_2^d} \{(x, y(\omega)) \mid x \in X \cap Q_1^{t(k)}, y(\omega) \in Y_L \cap Q_2^t\}. \tag{24}$$

We use the same rules as in the CPT algorithm [Chen et al., 2012] to construct the disjunctive set. $\mathcal{T}_2^d$ is maintained by a tree structure (called cutting plane tree). $\mathcal{T}_2^d$ contains all the nodes that do not have children nodes and is initialized with one global node defining the constraints for Q_2. If the solution $y^d(\omega)$ does not satisfy the mixed-integer restrictions, then the algorithm walks through the cutting plane tree to locate the deepest node from the root node that contains $y^d(\omega)$. Let us refer to this node as t_2^d. If $t_2^d \in \mathcal{T}_2^d$ (meaning it does not have any children node), two nodes are created as its children nodes and t_2^d is removed from $\mathcal{T}_2^d$. On the other hand, if $t_2^d \notin \mathcal{T}_2^d$, no new node is created. In this way, $\mathcal{T}_2^d$ is updated such that conditions (23) and (24) are satisfied (see also Algorithm 2).

Assuming that (24) is satisfied, we will use a cut generation linear program (CGLP) to derive a valid inequality. While the specific form of the CGLP is not critical to the proof of convergence of the methodology, a linear programming structure of the CGLP is important. The form of CGLP shown in (25) maximizes the depth of cut while restricting the one-norm of the coefficients to be 1 (see 25e). This version is similar to the CGLP used by ([Chen et al., 2012]), where the right-hand-side was fixed to be 1, and the one-norm of cut coefficients was minimized.

Using the partition (22), the CGLP shown in (25) can be formulated to derive multi-term disjunctive cuts.

$$\begin{aligned}
\max \quad & \pi_0(\omega) && \text{(25a)}\\
\text{s.t.} \quad & \pi_{2j}(\omega) = W_j^{t(k)}(\omega)^\top \lambda_{2t} + \mu_{2jt} - \nu_{2jt} \quad \forall j \in J, t \in \mathcal{T}_2^d && \text{(25b)}\\
& \pi_{1i}(\omega) = T_i^{t(k)}(\omega)^\top \lambda_{2t} + A_i{}^\top \lambda_1 + \mu_{1i} - \nu_{1i} \quad \forall i \in I, t \in \mathcal{T}_2^d && \text{(25c)}\\
& r^{t(k)}(\omega)^\top \lambda_{2t} + b^\top \lambda_1 + l_2^{t\top} \mu_{2t} + l_1^{t(k)\top} \mu_1 - u_2^{t\top} \nu_{2t} - u_1^{t(k)\top} \nu_1 \\
& \qquad \geq \pi_1(\omega)^\top x^k + \pi_2(\omega)^\top y^d(\omega) + \pi_0(\omega) \quad t \in \mathcal{T}_2^d && \text{(25d)}\\
& \sum_{i \in I} \alpha_{1i} + \sum_{j \in J} \alpha_{2j} = 1 && \text{(25e)}\\
& -\alpha_1 \leq \pi_1(\omega) \leq \alpha_1, -\alpha_2 \leq \pi_2(\omega) \leq \alpha_2 && \text{(25f)}\\
& \alpha_1, \alpha_2 \geq 0, \lambda_1, \lambda_{2t} \geq 0, \nu_1, \nu_{2t} \geq 0, \mu_1, \mu_{2t} \geq 0, \text{for } \forall\ t \in \mathcal{T}_2^d. && \text{(25g)}
\end{aligned}$$

The solution to the above CGLP provides the cut shown in (26).

$$\pi_1(\omega)^\top(x - x^k) + \pi_2(\omega)^\top(y(\omega) - y^d(\omega)) \geq \pi_0(\omega) \tag{26}$$

Note that (26) is associated with scenario ω, and we derive separate cuts for each $\omega \in \Omega$. Since (25) includes inequalities that are valid for $X \cap Q_1^{t(k)}$, this CGLP combines the convexification in the space $(x, y(\omega))$ where $x \in Q_1^{t(k)}$. As a result, the CGLP suggested in this paper is larger than that used for the original D^2 algorithm ([Sen and Higle, 2005]). However, the **ABC** algorithm will not require the convexification using the epi-reverse polar used in ([Sen and Higle, 2005]). In this sense, the cuts used in the **ABC** algorithm are different from those in the D^2 algorithm. The validity of such cuts is shown next.

Proposition 3. *Cutting plane (26) is valid for feasible set* $\{(x, y(\omega)) \mid T^{t(k)}(\omega)x + W^{t(k)}(\omega)y(\omega) \geq r^{t(k)}(\omega), x \in X \cap Q_1^{t(k)}, y(\omega) \in Y \cap Q_2\}$.

Proof. For any $\{(x, y(\omega)) \mid T^{t(k)}(\omega)x + W^{t(k)}(\omega)y(\omega) \geq r^{t(k)}(\omega), x \in X \cap Q_1^{t(k)}, y(\omega) \in Y \cap Q_2\}$, (25b,c,d) imply that

$$\pi_1^\top x + \pi_2^\top y(\omega) = (\lambda_{2t}^\top T^{t(k)}(\omega)x + \lambda_1^\top Ax + \mu_1^\top x - \nu_1^\top x) + (\lambda_{2t}^\top W^{t(k)}(\omega)y(\omega) + \mu_{2t}^\top y(\omega) - \nu_{2t}^\top y(\omega)) \tag{27a}$$

$$\geq r^{t(k)}(\omega)^\top \lambda_{2t} + b^\top \lambda_1 + {l_2^t}^\top \mu_{2t} + {l_1^{t(k)}}^\top \mu_1 - {u_2^t}^\top \nu_{2t} - {u_1^{t(k)}}^\top \nu_1 \tag{27b}$$

$$\geq \pi_1(\omega)^\top x^k + \pi_2(\omega)^\top y^d(\omega) + \pi_0(\omega), \tag{27c}$$

as required. □

The cut (26) is included in the second stage formulation, leading to a stronger approximation of the second stage polyhedron, and consequently, a stronger approximation of the value function, which is denoted as $f_L^{k,d}(x, \omega)$. In general, after d iterations of the CPT algorithm for the subproblem, we have

$$f_L^{k,d}(x, \omega) = \min \quad g(\omega)^\mathsf{T} y \tag{28a}$$

$$\text{s.t.} \quad W^{t(k)}(\omega)y \geq r^{t(k)}(\omega) - T^{t(k)}(\omega)x \tag{28b}$$

$$\Pi_2^d(\omega)y \geq \Pi_0^d(\omega) - \Pi_1^d(\omega)x \tag{28c}$$

$$y \in Y_L \cap Q_2 \tag{28d}$$

$$Y_L = \{y \geq 0, y \in \mathcal{R}^{n_2}\} \tag{28e}$$

where constraints (28c) denotes all the cuts generated during this round of subproblem solve. Depending on the course of the CPT algorithm, (28c) needs also be included in the CGLP to ensure convergence of the algorithm ([Chen et al., 2011]). After the approximation $f_L^{k,d}(x,\omega)$ is obtained, the cut-enhanced LP is re-solved and new cuts are generated as long as the inner iteration generate y^d that are fractional. At any outer iteration k, we allow at most D cuts to be added for the second stage CPT (approximation) algorithm. Once the process of solving the subproblem stops, we form a Benders' cut as in (13) and return to the master problem. During the implementation, we use sequence $\mathcal{G}_y^{t(k),\omega}$ to keep track of all CPT cuts generated for scenario ω with the first stage solutions from $Q_1^{t(k)}$ as well as cuts inherited from $t(k)$'s ancestor nodes. Cuts generated from first stage solutions x that do not belong to $Q_1^{t(k)}$ or one of its ancestors or for other scenarios are not included in (28b), e.g. by setting $\mathcal{G}_y^{t(k),\omega}[s] = 0$.

Given $x^k \in X \cap Q_1^{t(k)}$, the algorithm to solve subproblems ω with at most D cuts added is shown as Algorithm 2.

Algorithm 2 CPT-D

Initialize $d \leftarrow 1$, set CPT tree leaves set $\mathcal{T}_2^d(\omega) \leftarrow \{o\}$ where o is the root node with bounding constraint $Q_2^o \leftarrow Q_2$. Populate $W^{t(k)}(\omega)$, $r^{t(k)}(\omega)$, $T^{t(k)}(\omega)$ with valid cuts based on $\mathcal{G}_y^{t(k),\omega}$.
while $d \leq D$ **do**
  Evaluate $f_L^{k,d}(x,\omega)$ and get $y^d(\omega)$.
  if $y^d(\omega)$ satisfies mixed-integer restrictions **then**, STOP and $y^k(\omega) \leftarrow y^d(\omega)$
  else
    Find the deepest node σ that contains $y^d(\omega)$ in $\mathcal{T}_2^d$.
    if σ is a leaf node **then**,
      Make splits on σ and use updated $\mathcal{T}_2^d$ and first stage node bounds $Q_1^{t(k)}$ to formulate and solve (25) and obtain cut (26).
    else
      No splits are needed. Use σ and leaf nodes that in the subtree of σ and $Q_1^{t(k)}$ to formulate and solve (25) and obtain cut (26).
    end if
    Update $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ with the new cut.
  end if
  $d \leftarrow d+1$
end while
Extend $\mathcal{G}_y^{t(k),\omega}$ with the newly generated cuts (28c) by setting $\mathcal{G}_y^{t(k),\omega}[END+1...END+D] = 1$. For other $t \in \mathcal{T}_1 \backslash \{t(k)\}$, $\mathcal{G}_y^{t,\omega}[END+1...END+D] = 0$

Proposition 4. *Assume that the second stage problems are all bounded, and moreover, suppose that the cuts generated from (25) correspond to its extreme point solutions of the CGLP. Then for fixed $t(k)$ there exists a finite integer $D < \infty$, such that algorithm **CPT-D** solves all subproblems (2) indexed by ω to mixed-integer optimality; that is, with $x = x^k$ we have $f_L^{k,D}(x^k,\omega) = f(x^k,\omega)$ for $\forall\omega$.*

Proof. When D is large enough, algorithm **CPT-D** is the same as using CPT algorithm to solve each subproblem except that the CGLP is different. Due to the finiteness of the CPT algorithm proved in [Chen et al., 2011], all we need to prove is that for fixed $\mathcal{T}_2^k$ and $Q_1^{t(k)}$, only finitely many constraints

can be generated from the new CGLP (25) as $y^d(\omega)$ changes. To see this, note that each extreme point optimum of the CGLP is associated with a facet containing a specific combination of dual extreme points of the convex hull of the disjunctive set (in the space $y(\omega)$) [Balas, 1979]. Since dual constraints of the CGLP do not change with $y^d(\omega)$, and there are only finitely many dual extreme points to enumerate, and hence there are only finitely many constraints that can be generated from the CGLP (25). Ultimately, one therefore obtains all necessary facets of the disjunctive set in finitely many (inner) iterations indexed by d. Thus there exists a setting $D < \infty$ such that (2) is solved at scenario ω with x fixed at x^k, and hence, $f_L^{k,D}(x^k, \omega) = f(x^k, \omega)$. □

In our study, we will gradually increase the number of cuts we generate during any outer iteration k. To do so, we initialize $D \leftarrow 2$ and at each outer iteration, we increment $D \leftarrow D + 2$. This style of implementation is motivated by our prior experience (e.g. [Yuan and Sen, 2009]) that seeking very accurate objective function estimates requires much more computational resources than can be justified in early (outer) iterations of the algorithm; in later iterations however, seeking greater accuracy tends to pay off. This philosophy has proved to be effective in many settings, such as "inexact" line search in deterministic optimization and "stochastic decomposition" in stochastic programming ([Higle and Sen, 1994]).

3.2 Convexification implied by Branch-and-Bound

Our previous experience with similar SIP decomposition schemes have also revealed that cutting planes, by themselves, are often inadequate to the task of closing the duality gap in many realistic instances [Sen and Sherali, 2006, Yuan and Sen, 2009]. Similarly, most deterministic MIP algorithms combine valid inequalities in the context of Branch-and-Bound (B&B) methods, and as a result, Branch-and-Cut methods form the backbone for most state-of-the-art commercial solvers. The CPT algorithm of the previous section is a pure cutting plane method. The fact that it also utilizes a tree structure to manage the disjunctive sets inspires a way that transforms the B&B tree obtained from an MILP solver to help create a polyhedral approximation that gives the same IP optimal value as obtained from the B&B method. For the remainder of this section, we describe such an algorithm and prove that this approximation can also be obtained in finitely many steps. It turns out that this combination (of B&B with valid inequalities) will prove to be quite fruitful.

Suppose for fixed $x^k \in X \cap Q_1^{t(k)}$, subproblem $f(x^k, \omega)$ is either solved to optimality by a B&B method, or a truncated B&B process is carried out until a node limit or a time limit is reached. Let the optimal/incumbent solution be denoted $y^k(\omega)$. Due to the B&B (or truncated B&B) process, it is reasonable to assume that we have a set of leaf nodes. Let $\mathcal{T}_2^{\text{remain}}$ denote the remaining leaf nodes in the B&B tree and $\mathcal{T}_2^{\text{fathom}}$ denote the leaf nodes that have been fathomed. Then, we define $\mathcal{T}_2 =$

$\mathcal{T}_2^{\text{remain}} \cup \mathcal{T}_2^{\text{fathom}}$. Suppose the constraint set used in the calculation of $f_L(x^k, \omega)$ is given by

$$\mathcal{Y}_L(x^k, \omega) = \{y(\omega) | W(\omega)y(\omega) \geq r(\omega) - T(\omega)x^k, y(\omega) \in Y_L \cap Q_2\} \tag{29}$$

and define

$$\mathcal{Y}(x^k, \omega) = \mathcal{Y}_L(x^k, \omega) \cap Y \text{ and } \bigcup_{t \in \mathcal{T}_2} (\mathcal{Y}_L(x^k, \omega) \bigcap Q_2^t). \tag{30}$$

Since $Q_2^t, \ \forall \ t \in \mathcal{T}_2$ are disjoint from each other, $\mathcal{T}_2$ provides us a disjunctive relaxation in the space of $(x, y(\omega))$

$$\{X \cap Q_1^{t(k)}\} \times \mathcal{Y}_D(x^k, \omega) = \{X \cap Q_1^{t(k)}\} \times \bigcup_{t \in \mathcal{T}_2} (\mathcal{Y}_L(x^k, \omega) \bigcap Q_2^t). \tag{31}$$

Thus, the same form of CGLP as in (25) can be used to derive multi-term disjunctive cuts with replacing $\mathcal{T}_2^k$ with $\mathcal{T}_2$ in the formulation. The rest of algorithm will be similar to **CPT-D**. There are two phases: the first phase is to use a B&B method to either solve the second stage MIP, or a truncated B&B process discussed earlier. In either case, we have $\mathcal{T}_2$ which is used to obtain a disjunctive approximation. The second phase starts by seeking the value $f_L^{k,d}(x^k, \omega)$ with $d = 1$. At iteration d, if the optimal solution of $f_L^{k,d}(x^k, \omega)$ is fractional (denoted as $y^d(\omega)$), (25) is formulated based on $\mathcal{T}_2$ and $Q_1^{t(k)}$ to cut off $y^d(\omega)$. The new cut is encoded into $\mathcal{G}_y^{t(k)}$. Then $f_L^{k,d}(x^k, \omega)$ is re-evaluated and this process continues until $y^d(\omega)$ belongs to $\mathcal{Y}_D$. The method is shown in Algorithm 3.

Algorithm 3 BB-D

Initialize iteration $d \leftarrow 1$.
Phase 1:
Solve subproblem: Evaluate $f^k(x, \omega)$ by using a B&B method with $x = x^k$ for D iterations and get leaf nodes set $\mathcal{T}_2$ and solution $y^*(\omega)$.
Phase 2:
while true **do**
  Evaluate $f_L^{k,d}(x, \omega)$ and get $y^d(\omega)$.
  if $y^d(\omega) \in \mathcal{Y}_D$ **then**, STOP and $y^k(\omega) \leftarrow y^d(\omega)$
  else
    Use $\mathcal{T}_2$ and Q_1^t to formulate and solve (25) and obtain cut (26).
    Update $\Pi_1^d(\omega)$, $\Pi_2^d(\omega)$, $\Pi_0^d(\omega)$ with the new cut.
  end if
  $d \leftarrow d + 1$
end while Extend $\mathcal{G}_y^{t(k),\omega}$ to encode newly generated cuts.

While the form of the **BB-D** process is similar to the **CPT-D** process presented in the previous section, the inclusion of B&B, especially its truncated version, makes this decomposition approach much more realistic for practical instances of MIP in the second stage. Nevertheless, the proof of convergence derives from the same concept that one can obtain a polyhedral approximation of a disjunctive set embodied by a B&B tree. This is summarized in the following proposition.

Define **clconv** as the operation that yields the closure of the convex hull of a set and let the terminal iteration number of the **BB-D** process for outer iteration k be denoted d_k.

Proposition 5. *Under the same assumptions as in Proposition 4, Algorithm 3 terminates in finitely many steps. When D is sufficiently large so that the B&B process provides an optimal second stage solution, there exists $d < \infty$ such that for fixed $t(k)$, $f(x^k, \omega) = f_L^{k,d}(x^k, \omega)$. Here k is the iteration counter for the first stage, and d is the iteration counter for the second stage.*

Proof. Because $\mathcal{T}_2$ is fixed, and the extreme points of the CGLP have a one-to-one correspondence with the facets clconv$\{\mathcal{Y}_D(x^k, \omega)\}$ (see 31), it follows that in the worst case, all its facets are generated, and hence after finitely many iterations d we must have, $y^d \in$ clconv$\{\mathcal{Y}_D(x^k, \omega)\}$, which is the stopping rule. When D is sufficiently large (as in the second part of the proposition), the B&B tree defines a partition for which clconv$(\{\mathcal{Y}_D(x^k, \omega)\})$ has the same optimal value as the second stage MIP solution for any pair (x^k, ω). Hence $g(\omega)^\top y^d(\omega) \geq f(x^k, \omega)$. However, $g(\omega)^\top y^d(\omega) = f_L^{k,d}(x^k, \omega) \leq f(x^k, \omega)$, because the cuts are all valid. Hence $f(x^k, \omega) = f_L^{k,d}(x^k, \omega))$. □

Algorithm 2 and Algorithm 3 both use multi-term disjunctive cuts to obtain value function approximation. The difference between the two algorithms is the way to construct disjunctive sets. Algorithm 3 uses the B&B nodes to construct the disjunctive set, whereas, Algorithm 2 iteratively builds up the disjunctive set.

4 Illustrative Examples and a Computational Prototype

We begin this section by illustrating the workings of the algorithms of this paper via a generalization of an example that has been used by several authors, under alternative problem structures (e.g. fixed recourse ([Carøe and Schultz, 1998]), fixed tenders ([Ahmed et al., 2004]), and binary recourse variables ([Sen, 2003]). For the sake of completeness, we have included the Example 1.0 in the Appendix. That version only includes binary variables, and serves as a benchmark for the extensions solved below. In addition to illustrating the generality of our approach, we will also present computational results associated with a prototypical implementation using Matlab. One should recognize that in general, Matlab scripts cannot be expected to compete with CPLEX. Nevertheless, we show that as the number of scenarios increase, the Matlab code associated with our implementation performs better than the commercial solver, thus demonstrating the potential for the class of algorithms presented in this paper.

4.1 Illustrative Examples

Example 1.1 is an extension of **Example 1.0** (see Appendix), and is intended to illustrate the workings of the algorithm when we include general integer variables in the second stage.

$$\min \quad -1.5x_1 - 4x_2 + \sum_{\omega \in \Omega} p(\omega) f(x, \omega) \tag{32a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{32b}$$

where

$$f(x, \omega) = \min \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{33a}$$

$$\text{s.t.} \quad \begin{bmatrix} 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \\ 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \end{bmatrix} \leq r(\omega) - T(\omega)x \tag{33b}$$

$$y_i \in \{0, 1...5\}, i = 1, ..., 4; R \geq 0, \tag{33c}$$

$\Omega = \{\omega_1, \omega_2\}$, $p(\omega_1) = p(\omega_2) = 0.5$, $r(\omega_1) = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$, $T(\omega_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $r(\omega_2) = \begin{bmatrix} 10 \\ 3 \end{bmatrix}$, $T(\omega_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$.

The summary of applying **ABC** algorithm with **CPT-D** and **BB-D** on **Example 1.1** are shown in Table 1 and Table 2.

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Value function cut for Node
1	-M-5.5	Inf	(1,1)	1	-57	0	-76	1	$\eta \geq -73.7560 + 6.2292x_1 + 1.0268x_2$
2	-76.7292	-72	(0,1)	1	-57	0	-80.625	4	$\eta \geq -80.0714 + 0.8929x_1 + 11.2589x_2$
3	-73.7560	-72.8125	(0,0)	2	-60.2979	6	-80	5	$\eta \geq -70.1489 + 2.4734x_1 + 0.9468x_2$
4	-72.8125	-72	(0,1)	3	-57	0	-80	2	$\eta \geq -79 + 1.4583x_1 + 10.5x_2$
5	-72.5	-72.5	(0,1)	3					

Table 1: **ABC** Algorithm with **CPT-D** for Example 1.1

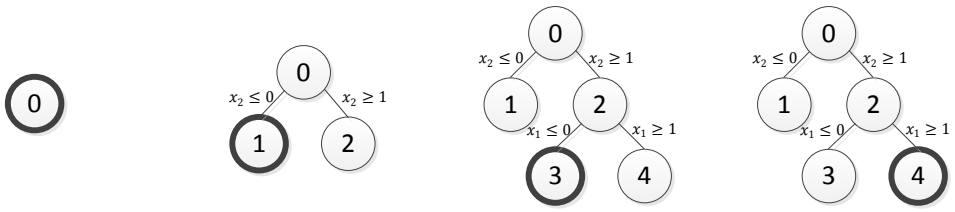
Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Value function cut for Node
1	-M-5.5	Inf	(1,1)	1	-57	0	-76	1	$\eta \geq -73.7560 + 6.2292x_1 + 1.0268x_2$
2	-76.7292	-72	(0,1)	1	-57	0	-80.1250	6	$\eta \geq -79.7232 + 1.5089x_1 + 11.1607x_2$
3	-73.7560	-72	(0,0)	2	-57	6	-80	4	$\eta \geq -68.5 + 2x_1$
4	-72.5625	-72	(0,1)	2	-57	0	-80	2	$\eta \geq -78.8571 + 1.4643x_1 + 10.3571x_2$
5	-72.5	-72.5	(0,1)	3					

Table 2: **ABC** Algorithm with **BB-D** for Example 1.1

In Table 1 and Table 2, the rows show the information generated in each successive iteration of the algorithm. The column header "Num Nodes" indicates the number of active nodes in the B&B tree and

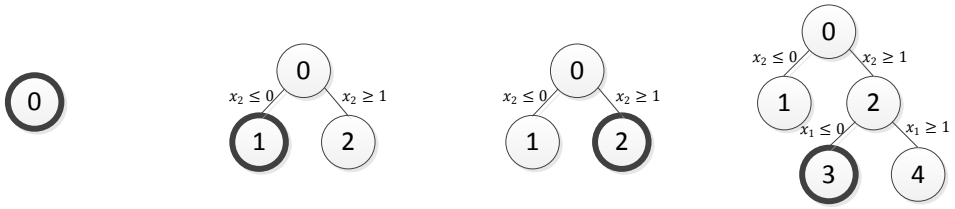
"Num Cuts" indicates the number of multi-term disjunctive cuts generated for that scenario. From the table, we can observe that both algorithms generate more cuts in the subproblem than in **Example 1.0** but differences between the execution of **BB-D** and **CPT-D** for this example are minimal.

To better illustrate the algorithm, we also include two sets of figures (Figure 1 and Figure 2) which show the master problem B&B tree at each iteration of the algorithm. The node with bold circle contains the solution x^k and gets value function updated in the iteration. Similar to the results shown in the tables, there are only minor differences for the master problem B&B tree between the two algorithms.



(a) Iteration 1,2 (b) Iteration 3 (c) Iteration 4 (d) Iteration 5

Figure 1: Master Problem B&B Tree for **ABC** Algorithm with **CPT-D** on Example 1.1

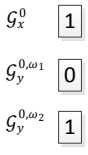


(a) Iteration 1,2 (b) Iteration 3 (c) Iteration 4 (d) Iteration 5

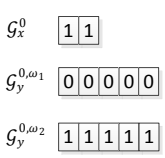
Figure 2: Master Problem B&B Tree for **ABC** Algorithm with **BB-D** on Example 1.1

In connection with this illustration, we also present two figures (Figure 3 and Figure 4) which show the encoded sequence for each master problem B&B tree node. With all generated Benders' cuts as one pool and all generated subproblem' cuts as another pool, the sequence encodes the validity of the cuts from each pool for any given node of the master problem B&B tree. Again, there are only minor differences in the execution of the two algorithms for this example.

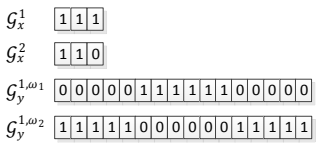
Example 1.2 This example extends **Example 1.1** by requiring general integer variables in the first stage as well. So instead of having x_1, x_2 to be binary, they are now allowed to be integers and bounded by $0 \le x_1, x_2 \le 5$. The summaries of applying two algorithms are shown in Table 3 and Table 4. Compared to the previous example, Tables -(3) and (4) demonstrate that due to a larger number of choices resulting from general integer variables in the first stage, the algorithms require more iterations



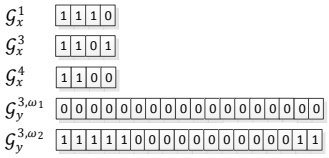
(a) Iteration 1



(b) Iteration 2

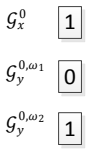


(c) Iteration 3

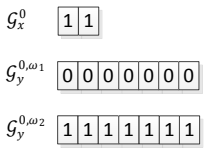


(d) Iteration 4

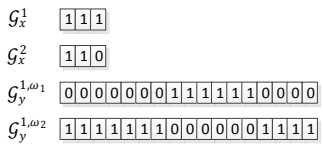
Figure 3: $\mathcal{G}_x, \mathcal{G}_y$ for **ABC** Algorithm with **CPT-D** on Example 1.1



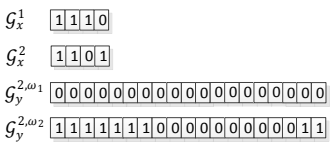
(a) Iteration 1



(b) Iteration 2



(c) Iteration 3



(d) Iteration 4

Figure 4: $\mathcal{G}_x, \mathcal{G}_y$ for **ABC** Algorithm with **BB-D** on Example 1.1

Iter	v	V	x	Num Nodes	$f(x,\omega_1)$	Num Cuts	$f(x,\omega_2)$	Num Cuts	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	100	0	-47	2	$\eta \geq -261 + 5x_1 + 52.5x_2$
2	-261	-1	(0,0)	1	-60.9545	4	-81.9048	4	$\eta \geq -71.4297 + 3.7564x_1 + 1.0352x_2$
3	-80.3241	-1	(0,3)	2	-19	0	-77.9165	6	$\eta \geq -83.2386 + 11.5934x_2$
4	-74.3945	-1	(0,1)	3	-57	0	-80	4	$\eta \geq -72.0974 + 1.1915x_1 + 3.5974x_2$
5	-72.5	-72.5	(0,1)	4					

Table 3: **ABC** Algorithm with **CPT-D** for Example 1.2

Iter	v	V	x	Num Nodes	$f(x,\omega_1)$	Num Cuts	$f(x,\omega_2)$	Num Cuts	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	100	0	-47	2	$\eta \geq -261 + 5x_1 + 52.5x_2$
2	-261	-1	(0,0)	1	-59.6667	5	-80	5	$\eta \geq -69.8333 + 2x_1 + 1.3333x_2$
3	-77.8333	-1	(0,3)	2	-19	0	-76	3	$\eta \geq -90.25 + 14.25x_2$
4	-73.6667	-59.5	(3,2)	5	-38	0	-61	5	$\eta \geq -74.1 + 1.8667x_1 + 9.5x_2$
5	-72.75	-62	(2,2)	5	-38	0	-66	6	$\eta \geq -81 + 5x_1 + 9.5x_2$
6	-72.5	-63	(0,1)	5	-57	0	-80	0	$\eta \geq -70.5 + 2x_1 + 2x_2$
7	-72.5	-72.5	(0,1)	5					

Table 4: **ABC** Algorithm with **BB-D** for Example 1.2

to solve the problem. Note also that using **BB-D** requires more iterations than **CPT-D** for this instance, although the average number of cuts for **BB-D** is fewer than that used by **CPT-D**.

Example 1.3 The instance we study below includes mixed-integer variables in both stages, and we also allow randomness in the T matrix, stated as follows.

$$T(\omega_1) = \begin{bmatrix} 0.1 & 0 \\ 0 & 0.5 \end{bmatrix}, T(\omega_2) = \begin{bmatrix} 0.3 & 0.2 \\ 0 & 0.2 \end{bmatrix}.$$

The summary of applying the algorithms are shown in Table 5 and Table 6. From Table (5) and (6)

Iter	v	V	x	Num Nodes	$f(x,\omega_1)$	Num Cuts	$f(x,\omega_2)$	Num Cuts	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	-22.8333	2	-66.5	2	$\eta \geq -91.1061 + 0.95x_1 + 8.3379x_2$
2	-93.8561	Inf	(5,0)	1	-59.3929	4	-71.3750	4	$\eta \geq -74.6060 + 1.8444x_1 + 0.4550x_2$
3	-81.6960	Inf	(0,2)	2	-57	0	-79.36	6	$\eta \geq -79.2 + 0.84x_1 + 5.51x_2$
4	-80.8424	Inf	(5,3)	2	-38	4	-66.925	8	$\eta \geq -65.1930 + 2.3312x_1 + 0.3581x_2$
5	-79.48	Inf	(5,2)	3	-57	0	-67.7086	10	$\eta \geq -84.1132 + 2.3273x_1 + 5.0612x_2$
6	-78.6816	Inf	(4,2)	4	-57	0	-69.2180	12	$\eta \geq -75.4437 + 0.5315x_1 + 5.1043x_2$
7	-78.16	Inf	(3,2)	4	-57	0	-70.5438	14	$\eta \geq -82.267 + 2.7281x_1 + 5.1554x_2$
8	-77.8543	Inf	(5,2)	6	-57	0	-66.4930	16	$\eta \geq -71.2465 + 4.75x_2$
9	-77.8065	Inf	(1,1)	7	-57	3	-76	13	$\eta \geq -72.3252 + 5.8252x_2$
10	-77.69	-72	(0,1)	7	-57	1	-77.3333	20	$\eta \geq -67.1667$
11	-77.5	-72	(2,2)	7	-57	0	-76	0	$\eta \geq -79.2524 + 1.1355x_1 + 5.2407x_2$
12	-77.5	-77.5	(2,2)	7					

Table 5: **ABC** Algorithm with **CPT-D** for Example 1.3

Iter	v	V	x	Num Nodes	$f(x,\omega_1)$	Num Cuts	$f(x,\omega_2)$	Num Cuts	Value function cut for Node
1	-M-27.5	Inf	(5,5)	1	-22.8333	2	-65.6250	3	$\eta \geq -91.2652 + 1.0375x_1 + 8.3697x_2$
2	-93.5777	Inf	(5,0)	1	-57	4	-70.5	4	$\eta \geq -73.6806 + 1.9861x_1$
3	-81.6806	Inf	(0,2)	2	-57	0	-76	4	$\eta \geq -77.2950 + 5.3975x_2$
4	-80.4686	-74.5	(5,3)	2	-38	6	-64.6375	5	$\eta \geq -65.8580 + 2.7049x_1 + 0.3382x_2$
5	-79.7361	-74.5	(4,2)	3	-57	0	-67.5333	7	$\eta \geq -73.1165 + 0.2569x_1 + 4.9111x_2$
6	-79	-74.5	(3,2)	4	-57	0	-68.4810	10	$\eta \geq -81.0190 + 3.7595x_1 + 3.5x_2$
7	-77.5098	-74.5	(5,2)	5	-57	0	-66	4	$\eta \geq -69.8442 + 1.0174x_1 + 1.6287x_2$
8	-77.5	-77	(2,2)	5	-57	0	-76	0	$\eta \geq -76.0454 + 1.1008x_1 + 3.6719x_2$
9	-77.5	-77.5	(2,2)	5					

Table 6: **ABC** Algorithm with **BB-D** for Example 1.3

we can conclude that **BB-D** requires far fewer disjunctive cuts on average for solving subproblems than **CPT-D**. In addition, the total number of iterations and Benders' cuts in the first stage are also fewer.

4.2 Experiments with a Computational Prototype.

In this subsection we report experiments with a Matlab prototype to give the reader a sense of the potential for the methodology presented in this paper. We designed a Matlab implementation of the **ABC** algorithm with calls the CLPEX LP solver whenever an LP solution is required. For all other purposes

(e.g. managing the B&B tree) the Matlab script operates in the Matlab environment. Because Matlab is a scripting language, there are large overheads in execution, and in general, cannot be expected to compete with codes written in C or C++. Such handicaps notwithstanding, we conducted an experiment to see how our procedures scale with increases in the number of scenarios. Moreover, we wish to study how a commercial software like CPLEX might perform on the same instances.

Three sets of instances are generated based on **Example 1.1-1.3**. For each example, we create 4,9,36,121,441,1681,l0201 scenarios by generating the right hand sides $r(\omega)$ from equidistant lattice points in $[5, 15] \times [5, 15]$ with equal probability assigned to each point. This methodology was borrowed from [Ahmed et al., 2004]. For the seven instances based on Example 1.3, we use the same random right hand side $r(\omega) = \begin{bmatrix} r_1(\omega) \\ r_2(\omega) \end{bmatrix}$ as in Example 1.2, but in addition $T(\omega)$ is also random. The entries for these matrices were 0 or 1 with equal probability.

Table 7 compares the performance of three algorithms: two based on using the **ABC** algorithm with **CPT-D** and **BB-D** whereas, the third algorithm used CPLEX 12.3 (with default setting) for the MILP formulation of the deterministic equivalent formulation (**DEF**). All approaches were run on a Windows PC operating with Intel i7-3770K 3.5GHz processors. Instances 1-3 in the table correspond to the variations based on Example 1.1-1.3. The column heading **Obj** denotes the optimal objective value of the SMIP, **Var** and **Constr** denote the number of variables and constraints in the **DEF**. The entries in column **ABC(CPT-D)** and **ABC(BB-D)** denote Iterations (Master Nodes, Second stage Leaf Nodes, Running Time) which correspond to the total number of iterations, the total number of nodes in the B&B tree in the master problem, the maximum leaf nodes encountered during solving the subproblem and the total running time. The **DEF** column shows the cpu time required to solve **DEF** using the default version CPLEX 12.3 MIP solver. The maximum cpu time allowed was 60 minutes.

The results reported in Table 7 clearly demonstrate that the approach of solving a **DEP** with a commercial solver does not scale well, failing to solve 9 instances for which the number of scenarios were somewhat large. In comparing the performance of **ABC(CPT-D)** and **ABC(BB-D)**, we observe that the former also runs into numerical difficulties for 4 of the larger instances. In contrast, **ABC(BB-D)** produces optimal solutions for all the instances within very reasonable computational times. From the log-log plot in Figure 5, as number of scenarios increases, the running time also increases polynomially for **ABC(BB-D)**. The slopes of each of the three graphs are slightly larger than one (with values 1.0048, 1.1245, 1.1677) which suggests that as the number of scenarios increases, the running time increases at a rate that is only slightly worse than "linear". To sum up, based on the instances tested, algorithm **ABC** shows very stable results and scales quite well with the number of scenarios.

*Obj=65.576 and the solution has numerical issues

	Scenarios	Obj	Var	Constr	ABC(CPT-D)	ABC(BB-D)	DEF
Instance 1	4	-63.50	26	17	7 (4, 5, 0.234)	5 (4,5,0.14)	0.23
	9	-66.56	56	37	6 (2, 6, 0.29)	18 (15, 10, 0.98)	0.02
	36	-66.83	218	145	7 (2, 7, 1.36)	6 (2, 7, 1.01)	0.02
	121	-67.17	728	485	7 (1, 8, 4.43)	6 (1, 8, 2.96)	0.16
	441	-65.58	2648	1765	16 (3, 15, 46.63)	10 (2, 7, 15.27)	1.58*
	1681	-64.72	10088	6725	16 (3, 17, 262.44)	12 (3, 23, 85.29)	Failed(>60mins)
	10201	-64.19	61208	40805	Numerical Issue	13 (3, 23, 583.08)	Failed(>60mins)
Instance 2	4	-63.50	26	17	12 (10, 10, 0.37)	12 (14, 10, 0.3)	0.02
	9	-66.56	56	37	20 (15, 10, 1.92)	18 (15, 10, 0.94)	0.02
	36	-69.86	218	145	19 (16, 12, 7.64)	18 (16, 10, 4.54)	0.03
	121	-71.12	728	485	18 (16, 11, 25.18)	17 (16, 11, 13.51)	4.09
	441	-69.64	2648	1765	20 (16, 22, 168.43)	18 (15, 21, 58.97)	Failed(>60mins)
	1681	-68.85	10088	6725	Numerical Issue	20 (16, 27, 312.36)	Failed(>60mins)
	10201	-68.45	61208	40805	Numerical Issue	21 (18, 31, 2333.307)	Failed(>60mins)
Instance 3	4	-63.50	26	17	10 (6, 7, 0.55)	19 (18,8,0.41)	0.00
	9	-64.22	56	37	19 (15, 10, 2.09)	19 (15, 10, 0.83)	0.13
	36	-66.42	218	145	25 (21, 12, 7.85)	25 (19, 13, 3.62)	4.18
	121	-64.78	728	485	25 (20, 12, 26.94)	25 (19, 13, 12.04)	Failed(>60mins)
	441	-63.33	2648	1765	28 (21, 24, 222.82)	27 (20, 24, 82.60)	Failed(>60mins)
	1681	-62.19	10088	6725	31 (22, 28, 1210.94)	28 (23, 33, 419.42)	Failed(>60mins)
	10201	-61.83	61208	40805	Numerical Issue	31 (24, 34, 3243.82)	Failed(>60mins)

Table 7: Comparison of **ABC** with DEF

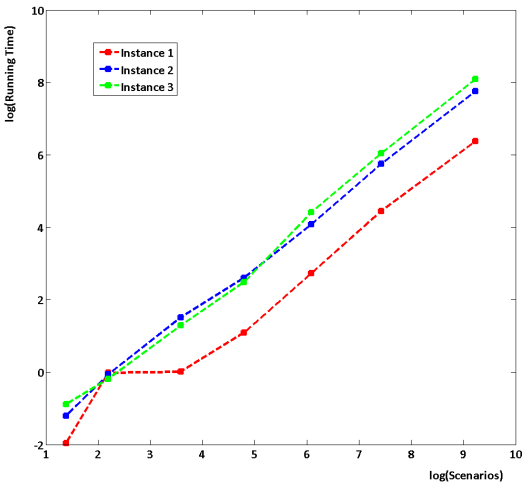


Figure 5: Log-log Plot of Running Time and Scenarios for **ABC** Algorithm with **BB-D** on 3 Instances

5 Conclusion

As stated at the outset, our paper returns to the general class of two stage SMIP problems that was the focus of the paper by [Carøe and Tind, 1998]. This class of problems involves mixed-integer variables in both stages, and randomness is also allowed in all data elements of the second stage MIP. Despite the elegance of the work of [Carøe and Tind, 1998], the chasm between first and second stage strategies has persisted over the past 15 years. Using several building blocks that have been effective in the interim, we have developed a time-staged decomposition algorithm for very general SMIP models. Other effective ideas, such as allowing the second stage problem to be solved inexactly are also permitted within the overall strategy. The key feature of this algorithm is a first stage B&B process (i.e. the ABC algorithm) which simultaneously guides both the construction of approximations as well as the search for optimal first stage decisions. Furthermore, the value function approximations remain piecewise linear and convex for each first stage B&B node, and similarly, the second stage relaxations (built using multi-term disjunctions) are also polyhedral. While these elements maintain convex building blocks, the overall search is facilitated by an encoding strategy (to record approximations) that allows us to approximate severe non-convexities typical of general SMIP models. This encoding mechanism is analogous to the way in which chromosomes are passed from generation to generation in nature.

We have presented a preliminary computational study in which a commercial LP solver was guided by a Matlab-based script, and this implementation was compared with a state-of-the-art MIP solver to obtain a solution for a deterministic extensive form SMIP. Our computations reveal that as the number of scenarios grow, our Matlab-guided implementation was faster and more stable than the commercial solver for an extensive form SMIP. We expect that future implementations using C/C++ programming will yield far greater capabilities.

Appendix: Example 1.0

Consider the following example that is shown in [Sen et al., 2003] which is a variation of an example from ([Schultz et al., 1998]).

$$\min \quad -1.5x_1 - 4x_2 + \sum_{\omega \in \Omega} p(\omega) f(x, \omega) \tag{34a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{34b}$$

where

$$f(x,\omega) = \min \quad -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R \tag{35a}$$

$$\text{s.t.} \quad \begin{bmatrix} 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \\ 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \end{bmatrix} \leq r(\omega) - T(\omega)x \tag{35b}$$

$$y_i \text{ binary } i = 1, ..., 4, R \geq 0, \tag{35c}$$

$\Omega = \{\omega_1, \omega_2\}$, $p(\omega_1) = p(\omega_2) = 0.5$, $r(\omega_1) = \begin{bmatrix} 5 \\ 2 \end{bmatrix}$, $T(\omega_1) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$, $r(\omega_2) = \begin{bmatrix} 10 \\ 3 \end{bmatrix}$, $T(\omega_2) = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$. We first apply the **ABC** algorithm with **CPT-D** to solve this example.

At iteration $k = 1$, the algorithm starts from solving the LP relaxed master problem with η bounded.

$$\min \quad -1.5x_1 - 4x_2 + \eta \tag{36a}$$

$$\text{s.t.} \quad x_1, x_2 \text{ binary} \tag{36b}$$

$$\eta \geq -M \tag{36c}$$

We get solution $(x_1, x_2, \eta) = (1, 1, -M)$ with objective $v = -M - 5.5$. Here only the root node is in the $B\&B$ tree. $x = (1, 1)$ with $Q_1^o = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 1\}$ is inserted into subproblems. Then **CPT-D** is called for each $\omega \in \Omega$.

For ω_1, $f_L(x, \omega_1)$ is solved and we get $y(\omega_1) = (0, 1, 0, 0.5, 0)$. y_4 is fractional and partitions are formed for integer variables: $\{y_4 \leq 0\} \cap Q_2$ or $\{y_4 \geq 1\} \cap Q_2$. The cut derived from CGLP for $x \in Q_1^o$ is

$$-2y_2 - 2y_4 + 2R \geq -4 + 2x_2. \tag{37}$$

After adding the cut, $f_L(x, \omega_1)$ is re-optimized and the solution is $y(\omega_1) = (0, 0, 0, 1, 0)$. It satisfies integer constraints. So no more cuts are generated in this iteration.

For ω_2, $f_L(x, \omega_2)$ is solved and we get $y(\omega_2) = (0, 1, 0, 0, 0)$. Again, this solution satisfies integer constraints, and hence, no cuts are needed. All scenarios have integer solution, so V is updated. $V = -29$.

The value function cut for $x \in Q_1^o$ is

$$-16.5x_2 + \eta \geq -40. \tag{38}$$

At iteration $k = 2$, the master problem continues to be solved by B&B method. We get solution $(x_1, x_2, \eta) = (1, 0, -40)$ with objective $v = -41.5$. Still only the root node is in the B&B tree. $x = (1, 0)$ with $Q_1^o = \{0 \le x_1 \le 1, 0 \le x_2 \le 1\}$ is inserted into subproblems. **CPT-D** is called for each $\omega \in \Omega$.

For ω_1, $f_L(x, \omega)$ is initialized as follows:

$$
\begin{aligned}
f_L(x, \omega_1) = \min \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R && \text{(39a)}\\
\text{s.t.} \quad & 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \le 5 - x_1 && \text{(39b)}\\
& 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \le 2 - x_2 && \text{(39c)}\\
& -2y_2 - 2y_4 + 2R \ge -4 + 2x_2 && \text{(39d)}\\
& 0 \le y_i \le 1 \ i = 1, ..., 4, R \ge 0 && \text{(39e)}
\end{aligned}
$$

where constraint (39d) is from cut (37) generated in iteration 1. (39) is solved and we get $y(\omega_1) = (0, 1, 0, 1, 0)$. It satisfies integer constraints. So no cuts are needed.

For ω_2, $f_L(x, \omega_2)$ is solved and we get $y(\omega_2) = (0.1154, 1, 0, 0.1538, 0)$. y_1 is the variable we choose to split. The partitions we form are: $\{y_1 \le 0\} \cap Q_2$ or $\{y_1 \ge 1\} \cap Q_2$. The cut derived from CGLP for $x \in Q_1^o$ is

$$-4.875y_2 - 6.5y_4 + 1.625R \ge -6.5 + 1.625x_1. \tag{40}$$

After adding the cut, $f_L(x, \omega_2)$ is re-optimized and the solution is

$$y(\omega_2) = (0.056, 1, 0.222, 0, 0). \tag{41}$$

Since $y(\omega_2)$ is located on the root node, no more splits are needed. The same partition is used: $\{y_1 \le 0\} \cap Q_2$ or $\{y_1 \ge 1\} \cap Q_2$. The cut derived from CGLP for $x \in Q_1^o$ is

$$-2.25y_2 - 4.5y_3 + 2.25R \ge -2.25. \tag{42}$$

After adding the cut, $f_L(x, \omega_2)$ is re-optimized and the new solution is $y(\omega_2) = (0.06, 0.68, 0.16, 0.24, 0)$. Again, $y(\omega_2)$ is located on the root node, and no more splits are needed. The same partition: $\{y_1 \le 0\} \cap Q_2$ or $\{y_1 \ge 1\} \cap Q_2$ is used to formulate CGLP. With only $y(\omega_2)$ changed, another cut derived from CGLP for $x \in Q_1^o$ is

$$-2.5y_3 - 2.5y_4 + 2.5R \ge -2.5 + 2.5x_1. \tag{43}$$

After adding the cut, $f_L(x, \omega_2)$ is re-optimized and the solution is $y(\omega_2) = (0.1667, 1, 0, 0, 0)$. $y(\omega_2)$ is located still on the root node. No more splits are needed. The same partition: $\{y_1 \le 0\} \cap Q_2$ or

$\{y_1 \geq 1\} \cap Q_2$ is used to formulate CGLP. The cut derived from CGLP for $x \in Q_1^o$ is

$$-6y_1 + 1.5R \geq 0. \tag{44}$$

After adding the cut, $f_L(x, \omega_2)$ is re-optimized and the solution is $y(\omega_2) = (0, 1, 0, 0, 0)$. It satisfies integer constraints. So no more cuts are needed. All scenarios have integer solution, so V is updated. $V = -34.5$

The value function cut for $x \in Q_1^o$ is

$$-7.55x_1 - 3.8333x_2 + \eta \geq -40.55. \tag{45}$$

At iteration $k = 3$, the master problem continues to be solved by the B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -40)$ with objective $v = -40$. The solution is on node 1 with $Q_1^1 = \{0 \leq x_1 \leq 0, 0 \leq x_2 \leq 1\}$. Hence $x = (0, 0)$ and Q_1^1 are treated as input parameters for **CPT-D** for each $\omega \in \Omega$.

For ω_1, 0 cuts are needed. The solution is $y(\omega_1) = (0, 1, 0, 1, 0)$.

For ω_2, 1 cut is needed (shown below). The solution is $y(\omega_1) = (0, 0, 0, 1, 0)$.

$$-3.6923y_2 - 2.4615y_3 - 3.6923y_4 + 1.2308R \geq -3.6923. \tag{46}$$

V is updated. $V = -37.5$ and the value function cut for Q_1^1 is

$$-8.3333x_2 + \eta \geq -37.5 \tag{47}$$

At iteration $k = 4$, with updated value function cut for node 1, the master problem continues to be solve by B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. $V - v \leq \epsilon$. The algorithm stops

A short summary of using **ABC** algorithm with **CPT-D** to solve this problem is shown in Table 8.

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Value function cut for Node
1	-M-5.5	inf	(1,1)	1	-28	1	-19	0	$\eta \geq -40 + 16.5x_2$
2	-41.5	-29	(1,0)	1	-47	0	-19	4	$\eta \geq -40.55 + 7.55x_1 + 3.8333x_2$
3	-40	-34.5	(0,0)	2	-47	0	-28	1	$\eta \geq -37.5 + 8.3333x_2$
4	-37.5	-37.5	(0,0)	2					

Table 8: **ABC** Algorithm with **CPT-D** for Example 1.0

Each row shows the information for one iteration. Column "Num Nodes" denotes the number of active nodes in the B&B tree. "Num Cuts" means the number of multi-term disjunctive cuts generated for that scenario.

We also apply the same **ABC** algorithm but with **BB-D** to solve this example. The algorithm starts

from solving the LP relaxed master problem with η bounded.

At iteration $k = 1$, We get solution $(x_1, x_2, \eta) = (1, 1, -M)$ with objective $v = -M - 5.5$ from the B&B method. $x = (1, 1)$ with $Q_1^o = \{0 \le x_1 \le 1, 0 \le x_2 \le 1\}$ is inserted into **BB-D** for each $\omega \in \Omega$.

For ω_1, $f(x, \omega_1)$ is solved by the B&B method and we get 2 nodes in $\mathcal{T}_2$. With 1 cut derived from CGLP for $x \in Q_1^o$:

$$-2y_2 - 2y_4 + 2R \ge -4 + 2x_2, \tag{48}$$

the solution falls into $\mathcal{T}_2$ and no more cuts are needed. $y(\omega_1) = (0, 0, 0, 1, 0)$.

For ω_2, $f(x, \omega_2)$ is solved by B&B method and we get 1 nodes in $\mathcal{T}_2$. No cuts are needed. $y(\omega_2) = (0, 1, 0, 0, 0)$. All scenarios have integer solution, so V is updated. $V = -29$

The value function cut for $x \in Q_1^o$ is

$$-16.5x_2 + \eta \ge -40 \tag{49}$$

At iteration $k = 2$, the master problem continues to be solved by B&B method. We get solution $(x_1, x_2, \eta) = (1, 0, -40)$ with objective $v = -41.5$. $x = (1, 0)$ with $Q_1^o = \{0 \le x_1 \le 1, 0 \le x_2 \le 1\}$ is inserted into subproblems. **CPT-D** is called for each $\omega \in \Omega$.

For ω_1, $f(x, \omega_1)$ is initialized as follows:

$$\begin{aligned}
f(x, \omega_1) = \min \quad & -16y_1 - 19y_2 - 23y_3 - 28y_4 + 100R && \text{(50a)}\\
\text{s.t.} \quad & 2y_1 + 3y_2 + 4y_3 + 5y_4 - R \le 5 - x_1 && \text{(50b)}\\
& 6y_1 + 1y_2 + 3y_3 + 2y_4 - R \le 2 - x_2 && \text{(50c)}\\
& -2y_2 - 2y_4 + 2R \ge -4 + 2x_2 && \text{(50d)}\\
& y_i \text{ binary } i = 1, ..., 4, R \ge 0 && \text{(50e)}
\end{aligned}$$

where constraint (50d) is from cut (48) generated in iteration 1. (50) is solved by B&B method and we get $\mathcal{T}_2$ with 1 node. So no cuts are needed. $y(\omega_1) = (0, 1, 0, 1, 0)$.

For ω_2, $f(x, \omega_2)$ is solved by B&B method and we get $\mathcal{T}_2$ with 4 nodes. 4 cuts are derived from CGLP for $x \in Q_1^o$:

$$\begin{aligned}
-8.6667y_1 + 2.1667R &\ge 0\\
-4y_3 + 4R &\ge 0\\
-3.75y_2 - 5y_4 + 1.25R &\ge -5\\
-x_1 - y_4 + R &\ge -1
\end{aligned} \tag{51}$$

With 4 cuts added, the solution $y(\omega_1) = (0, 1, 0, 1, 0)$.

All scenarios have integer solution, so V is updated. $V = -34.5$

The value function cut for $x \in Q_1^o$ is

$$-4.5x_1 - 3.8333x_2 + \eta \geq -37.5. \tag{52}$$

At iteration $k = 3$, the master problem continues to be solve by B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. The solution is on node 2 with $Q_1^1 = \{0 \leq x_1 \leq 1, 0 \leq x_2 \leq 0\}$. $x = (0, 0)$ and Q_1^2 are treated as input parameters for **BB-D** for each $\omega \in \Omega$.

For ω_1, 0 cuts are needed. The solution is $y(\omega_1) = (0, 1, 0, 1, 0)$.

For ω_2, 0 cuts are needed. The solution is $y(\omega_1) = (0, 0, 0, 1, 0)$. V is updated. $V = -37.5$ and the value function cut for Q_1^2 is

$$-2.8333x_1 - 5.1667x_2 + \eta \geq -37.5 \tag{53}$$

At iteration $k = 4$, with updated value function cut for node 2, the master problem continues to be solve by B&B method. We get solution $(x_1, x_2, \eta) = (0, 0, -37.5)$ with objective $v = -37.5$. $V - v \leq \epsilon$. The algorithm stops

A short summary of using **ABC** algorithm with **BB-D** to solve this problem is shown in Table 9. As

Iter	v	V	x	Num Nodes	$f(x, \omega_1)$	Num Cuts	$f(x, \omega_2)$	Num Cuts	Value function cut for Node
1	-M-5.5	inf	(1,1)	1	-28	1	-19	0	$\eta \geq -40 + 16.5x_2$
2	-41.5	-29	(1,0)	1	-47	0	-19	4	$\eta \geq -37.5 + 4.5x_1 + 3.8333x_2$
3	-37.5	-34.5	(0,0)	2	-47	0	-28	0	$\eta \geq -37.5 + 2.8333x_1 + 5.1667x_2$
4	-37.5	-37.5	(0,0)	2					

Table 9: **ABC** Algorithm with **BB-D** for Example 1.0

you may notice, there is only small difference between **BB-D** and **CPT-D** for **Example 1**. At iteration 2, because **BB-D** uses partitions with 4 terms to generate cuts from the beginning, the quality of the cut is better than **CPT-D** which results in no more cuts needed for subproblems in the following iterations. But CGLP with 4 terms take more time to solve.

References

[Ahmed et al., 2004] Ahmed, S., Tawarmalani, M., and Sahinidis, N. V. (2004). A finite branch-and-bound algorithm for two-stage stochastic integer programs. *Mathematical Programming*, 100(2):355–377.

[Balas, 1979] Balas, E. (1979). Disjunctive programming. *Annals of Discrete Mathematics*, 5:3–51.

[Balas et al., 1993] Balas, E., Ceria, S., and Cornuéjols, G. (1993). A lift-and-project cutting plane algorithm for mixed 0-1 programs. *Mathematical Programming*, 58:295–324.

[Carøe and Tind, 1998] Carøe, C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83:451–464.

[Carøe and Schultz, 1997] Carøe, C. C. and Schultz, R. (1997). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24:37–45.

[Carøe and Schultz, 1998] Carøe, C. C. and Schultz, R. (1998). A two-stage stochastic program for unit commitment under uncertainty in a hydro-thermal power system. In *Preprint SC 98-11, Konrad-Zuse-Zentrum fur Informationstechnik*, pages 98–13.

[Chen et al., 2011] Chen, B., Küçükyavuz, S., and Sen, S. (2011). Finite disjunctive programming characterizations for general mixed integer linear programs. *Operations Research*, 59(1):202–210.

[Chen et al., 2012] Chen, B., Küçükyavuz, S., and Sen, S. (2012). A computational study of the cutting plane tree algorithm for general mixed-integer linear programs. *Operations Research Letters*, 40(1):15 – 19.

[Escudero et al., 2007] Escudero, L., Garn, A., Merino, M., and Pérez, G. (2007). A two-stage stochastic integer programming approach as a mixture of branch-and-fix coordination and benders decomposition schemes. *Annals of Operations Research*, 152:395–420. 10.1007/s10479-006-0138-0.

[Gade et al., 2012] Gade, D., Küçükyavuz, S., and Sen, S. (2012). Decomposition algorithms with parametric gomory cuts for two-stage stochastic integer programs. *Mathematical Programming*, pages 1–26.

[Higle and Sen, 1994] Higle, J. L. and Sen, S. (1994). Finite master programs in regularized stochastic decomposition. *Mathematical Programming*, 67(1-3):143–168.

[Kong et al., 2006] Kong, N., Schaefer, A. J., and Hunsaker, B. (2006). Two-stage integer programs with stochastic right-hand sides. *Mathematical Programming*, 108(24):275–296.

[Louveaux and Schultz, 2003] Louveaux, F. and Schultz, R. (2003). Stochastic integer programming. *Handbooks in Operations Research and Management Science*, 10:213–266.

[Lulli and Sen, 2004] Lulli, G. and Sen, S. (2004). A branch-and-price algorithm for multistage stochastic integer programming with application to stochastic batch-sizing problems. *Management Science*, 50(6):786–796.

[Ntaimo, 2010] Ntaimo, L. (2010). Disjunctive decomposition for two-stage stochastic mixed-binary programs with random recourse. *Operations Research*, 58(1):229–243.

[Owen and Mehrotra, 2001] Owen, J. H. and Mehrotra, S. (2001). A disjunctive cutting plane procedure for general mixed-integer linear programs. *Mathematical Programming*, 89(3):437–448.

[Schultz et al., 1998] Schultz, R., Stougie, L., and van der Vlerk (1998). Solving stochastic programs with integer recourse by enumeration: A framework using gröbner basis. *Mathematical Programming*, 83(1):229–252.

[Sen, 2003] Sen, S. (2003). Algorithms for stochastic mixed-integer programming models. *Handbooks in Operations Research and Management Science*, 12:515–558.

[Sen, 2010] Sen, S. (2010). *Stochastic Mixed-Integer Programming Algorithms: Beyond Benders' Decomposition*. John Wiley & Sons, Inc.

[Sen and Higle, 2005] Sen, S. and Higle, J. L. (2005). The c3 theorem and a d2 algorithm for large scale stochastic mixed-integer programming: Set convexification. *Mathematical Programming*, 104(1):1–20.

[Sen et al., 2003] Sen, S., Higle, J. L., and Ntaimo, L. (2003). A summary and illustration of disjunctive decomposition with set convexification. *Woodruff, D. (ed.) Network Interdiction and Stochastic Integer Programming*, pages 105–125.

[Sen and Sherali, 2006] Sen, S. and Sherali, H. D. (2006). Decomposition with branch-and-cut approaches for two-stage stochastic mixed-integer programming. *Mathematical Programming*, 106(2):203–223.

[Sherali and Fraticelli, 2002] Sherali, H. D. and Fraticelli, B. M. (2002). A modification of benders' decomposition algorithm for discrete subproblems: An approach for stochastic programs with integer recourse. *Journal of Global Optimization*, 22:319–342.

[Sherali and Zhu, 2006] Sherali, H. D. and Zhu, X. (2006). On solving discrete two-stage stochastic programs having mixed-integer first- and second-stage variables. *Mathematical Programming*, 108(2-3):597–616.

[Trapp et al., 2012] Trapp, A. C., Prokopyev, O. A., and Schaefer, A. J. (2012). On a level-set characterization of the integer programming value function and its application to stochastic programming. *Operations Research*.

[Wollmer, 1980] Wollmer, R. D. (1980). Two-stage linear programming under uncertainty with $0-1$ integer first stage variables. *Mathematical Programming*, 19(3):279–288.

[Yuan and Sen, 2009] Yuan, Y. and Sen, S. (2009). Enhanced cut generation methods for decomposition-based branch and cut for two-stage stochastic mixed-integer programs. *INFORMS Journal on Computing*, 21(3):480–487.